$1.95

QUICK REFERENCE CARD

Atari BASIC And OSS BASIC A+

This card provides a complete syntax summary of all statements and functions in both Atari BASIC and OSS BASIC A+. The various keywords of the languages are grouped as follows:

First:	by category, with a heading for each group. A keyword may appear in more than one category.
Second:	within the category group, those keywords found in both BASICs precede those found only OSS BASIC A+.
Third:	within each language partition, all statements precede all functions. Functions are denoted by an 'f' in front of the keyword.
Finally:	within each list of statements and list of functions, keywords are placed alphabetically.

NOTE: All capabilities found in OSS BASIC A+ are shown shaded, as in this sentence.

DEFINITION OF TERMS

KEYWORDS are shown in bold face type, and should be typed as shown. The following syntax for each keyword is shown in normal type and generally consists of zero or more of the syntax items shown below. Explanations are shown in italics.

Items enclosed [in square brackets] are optional. Enclosed items [followed by ellipses ...] may be repeated any number of times.

SYNTAX ITEMS

<stmt>	any valid statement
var	any VARiable
avar	an Arithemetic var
svar	a String var
mvar	a Matrix var (or matrix element)
asvar	avar or svar, but never mvar
filename	a sexp used as a file specifier
pm	aexp used as P/M #
addr	an aexp used as a memory address
<stmts>	any number of valid statements placed on any number of lines
exp	any expression
aexp	an Arithmetic exp
sexp	a String exp
line	aexp used as a line #
fn	aexp used as a file #

Legal forms of file specifiers: <device>:<file>.<ext> where <device> consists of a single letter optionally followed by a single digit. When the device is the disk, <file> is any name consisting of 1 to 8 alphanumeric characters, the first of which is a letter. <ext> is an optional 1 to 3 alphanumeric characters. Here are some examples:

E: (the screen editor)
P: (the printer)
R2: (RS-232 port number 2)
D2:MENU. SAV (a disk file on drive 2 with the name "MENU" and the name extension "SAV")

product #
BREF 1.0

COMMAND & CONTROL

BYE
goes to memo pad
CLR
zeroes simple variables, changes all DIMs to 0
CLOAD
load a program from cassette
CSAVE
save a program to cassette
ENTER filename
only works with ATASCII version of a program (see LIST); actually a merge unless NEW is used first
LIST file name
lists program to file in ATASCII just as it appears on the screen for LIST alone
LOAD filename
load a previously SAVEd program
RUN filename
load and run a SAVEd program
SAVE filename
save a program to a file using internal format

PROGRAM DEVELOPMENT STATEMENTS

CLOAD
load a program from cassette
CONT
continue a program after a STOP or BREAK
CSAVE
save a program to cassette
END
close all files, stop the program
ENTER file name
merges an ATASCII (LISTed) program into that already in memory
LIST [filename]
list program in ATASCII to screen or file
LIST [filename,] line [,line]
list only a portion of a program
LOAD filename
load a previously SAVEd program
NEW
remove all programs and variables from memory
REM <any remark>
allows commenting of program listings
RUN
begin executing program in memory at lowest line number
RUN [filename]
load a SAVEd program and start executing it
SAVE filename
save a program in memory to a file in internal format
STOP
halt execution of program
f FRE(0)
returns amount of memory still available

DEL line [,line]
delete all lines in range specified
LOMEM addr
can reserve memory; does a NEW
LVAR filename
list all variables in use by program in memory to given file
RENUM [start][, increment]
renumbers entire program
TRACE
begin displaying each line's number as it is executed
TRACEOFF
cease displaying line numbers

PROGRAM CONTROL

END
close files, stop program

FOR avar = aexp TO aexp [STEP aexp] <stmts>:
NEXT avar
traditional loop control

GOSUB line
call a subroutine

GOTO line
transfer control to new line

IF aexp **THEN** <stmt>[:<stmt>...]
statements after THEN are executed only if the aexp is non-zero

IF aexp **THEN** line
control is transferred to new line only if the aexp is non-zero

NEXT {*see FOR*}

ON aexp **GOTO** line [,line ...]
ON aexp **GOSUB** line [,line ...]
if aexp = 1, control moves to first line given; if aexp = 2, then to a second line; etc.

POP
removes last FOR, GOSUB, or WHILE from stack

RETURN
end of subroutine called by GOSUB

RUN [filename]
start program from beginning

STEP {*see FOR*}

STOP
halts program, allows CONT

THEN {*see IF above*}

TO {*see FOR*}

TRAP line
if a subsequent error occurs, control is transferred to line specified

CONT
after a TRAPped error, continue at line after error

ELSE {*see IF below*}

ENDIF {*see IF below*}

ENDWHILE {*see WHILE*}

IF aexp : <stmts>
[ELSE : <stmts>]
ENDIF
use when both 'true' and 'false' paths are needed; may be nested 127 deep

WHILE aexp:
<stmts>
ENDWHILE
loops between WHILE and ENDWHILE so long as aexp is non-zero

f ERR (aexp)
returns last run-time error code

CONSOLE & FILE I/O

CLOSE #fn
cease I/O to file channel fn

GET #fn, avar
set a single byte from fn

INPUT [#fn,] asvar [,asvar ...]
input ATASCII data

LPRINT [exp [;exp ...] [, exp ...]]
output ATASCII to line printer

OPEN #fn, mode, avar, filename
begin I/O with filename on channel fn

NOTE #fn, avar, avar
find current position/disk file

POINT #fn, avar, avar
change current file position

PRINT [#fn]
output new line only

PRINT exp [[; exp ...] [,exp ...]] [;]
output data items in ATASCII

PRINT #fn [[; exp...] [,exp ...]] [;]
output ATASCII items to a file

PUT #fn, aexp
output a single byte to fn

STATUS #fn, avar
dynamic status check

XIO aexp, #fn, aexp, aexp, filename
extended I/O operation

? {*same as PRINT*}
usable wherever PRINT is legal

CONSOLE & FILE I/O (cont)

BGET #fn, addr, len
set binary block from file fn

BPUT #fn, addr, len
put a binary block to file fn

INPUT "...", var [,var ...]
allows prompt to replace "?"

LPRINT [#fn,] USING sexp, [exp[,exp ...]][;]
see special table:PRINT USING

PRINT [#fn,] USING sexp, [exp[,exp ...]] [;]
see special table:PRINT USING

RGET #fn, asvar [,asvar ...]
get data items in special record-oriented format

RPUT #fn, exp [, exp ...]
put data items in special record-oriented format

TAB [#fn,] aexp
move to given print column

f **TAB** (aexp)
function version only usable in a PRINT stmt

MACHINE CONTROL

MOVE fromaddr, toaddr, lenaexp
move any piece of memory to anywhere; moves "down" if lenaexp is positive (contracts); moves "up" if lenaexp is negative (expands)

POKE addr, aexp
change contents of memory location addr to aexp

f **PEEK** (addr)
returns contents of memory location addr

f **USR** (addr [,aexp ...])
calls user assembly language subroutine at addr

DPOKE addr, aexp
change contents of WORD at location addr

f **DPEEK** (addr)
returns contents of WORD at location addr

OPERATOR PRECEDENCE TABLE

The operators of BASIC are listed in order precedence, from highest to lowest. Higher precedence implies the operator will be executed first. Example: 3+4x5 is seen as 3+ (4*5) because '*' has a higher precedence than '+'.

( ) functions ()	parenthesized subexpressions
=<>><>=<=	string comparisons [e.g., A$<> "EXIT"]
NOT + –	unary operators only [e.g., –3*Z]
Λ	exponentiation
& !	binary "and", binary "or"
* /	multiply and divide
+ –	add and subtract
=<>><>=<=	numeric comparisons [e.g., TOTAL > 30]
AND	logical "and" (always gives 1 or 0 result)
OR	logical "or" (always gives 1 or 0 result)
,	when used in array and function references [e.g., PRINT ARRAY (7,5)]

NOTE: In Atari BASIC, NOT was given a precedence just above AND, but it does not always execute properly unless it is followed by a sub-expression in parentheses [e.g., NOT (A>B) is safe].

ASSIGNMENT & MATHEMATICS

[LET] avar = aexp
[LET] mvar = aexp
arithmetic assignment; keyword is optional
DEG
selects degrees for trig functions
RAD
selects radians for trig functions
f **ABS** (aexp)
returns absolute value of argument aexp
f **ATN** (aexp)
returns arc tangent of argument; returns radians or degrees, as selected
f **CLOG** (aexp)
returns common log (base 10) of argument
f **COS** (aexp)
returns cosine of argument
f **EXP** (aexp)
returns 'e' to the power aexp, 'exponentiation'
f **INT** (aexp)
returns largest integer less than or equal to argument
f **LOG** (aexp)
returns natural logarithm of the argument
f **RND (0)**
returns a pseudo-random number between 0 (inclusive) and 1 (exclusive)
f **SGN** (aexp)
returns +1, 0, −1 according to the sign of the argument (0 only if argument is 0)
f **SIN** (aexp)
returns sine of argument
f **SQR** (aexp)
square root of argument
f **VAL** (sexp)
returns the 'value' of a number contained in a string

INITIALIZATION

CLR
zeros numeric variables, sets all DIMs to zero
DEG
selects degrees for trig functions
DIM svar (aexp)
DIM mvar (aexp[,aexp])
allocate space for either a string or array
RAD
selects radians for trig functions
f **FRE (0)**
returns amount of memory still available

LOMEM addr
can reserve memory; does a NEW
SET aexp, aexp
see separate chart
f **SYS** (aexp)
returns value SET before

DOS COMMANDS

DOS
exit to "DOS"

CP
same as DOS
DIR filename
list disk directory on screen
ERASE filename
remove file from disk
PROTECT filename
disallow writes and/or erases of given filename
RENAME filenames
changes name of a file—CAUTION: form must be "Dn: oldname, newname"
UNPROTECT filename
remove file protection

STRING & CHARACTER HANDLING

[LET] svar = sexp
the destination string variable may be subscripted
f ADR (svar)
returns the address of the given string
f ASC (sexp)
returns numeric value of first byte of given string
f CHR$ (aexp)
returns a one byte string—character has a value of aexp
f LEN (sexp)
returns length of string
f STR$ (aexp)
returns a string equivalent to what would be visible if aexp were PRINTed

[LET] svar = sexp [,sexp ...]
allows concatenation of several strings
f FIND (sexp, sexp, aexp)
finds location of 2nd str within 1st string starting at given position plus one

GRAPHICS, SOUND, & PLAYER/MISSILE GRAPHICS

COLOR aexp
choose a color for subsequent PLOT and DRAWTO
DRAWTO aexp, aexp
draw a line from last point PLOTted or drawn to
GRAPHICS aexp
choose a graphics mode
LOCATE aexp, aexp, avar
find what color a given point on the screen is
PLOT aexp, aexp
plot a single point (pixel)
POSITION aexp, aexp
set screen location cursor
SETCOLOR aexp, aexp, aexp
change color register values; order is register number, hue, luminance
SOUND aexp, aexp, aexp, aexp
change sound register values; order is register number, frequency, waveform, volume
f PADDLE (aexp)
get current paddle value
f PTRIG (aexp)
returns 0 if trigger pushed
f STICK (aexp)
get current joystick position
f STRIG (aexp)
returns 0 if trigger pushed

MISSLE pm, aexp, aexp
"shoot" a missile
PMCLR pm
clear a player area
PMCOLOR pm, aexp, aexp
change a player color—same format as SETCOLOR
PMGRAPHICS aexp
select player/missile mode
PMMOVE pm[,aexp] [;aexp]
move a player or missile
PMWIDTH pm, aexp
change player/missile width
f BUMP (pmnum, aexp)
check for player/missile and/or playfield collisions
f HSTICK (aexp)
returns −1, 0, +1 if joystick is left, center, right
f PEN (aexp)
returns light pen values
f PMADR (pm)
gets address of a player or missile
f VSTICK (aexp)
returns −1, 0, +1 if joystick is down, center, up

IN-MEMORY DATA HANDLING

DATA <ATASCII data>
data may contain any characters except a comma

READ asvar [,asvar ...]
evaluate next data from DATA statement(s) and place in specified variable

RESTORE [line]
move data pointer to given line number, (or beginning of program)

DATA ["<quoted data>"]
[<ATASCII data>]
if data is quoted may contain any characters except another quote

READ var [,var ...]
may read directly into subscripted array elements or substrings

BASIC ERROR MESSAGES

Number	Message	Number	Message
1	Break Key Abort	16	RETURN With No Matching GOSUB
2	Memory Full	17	Bad Line (syntax error/line)
3	Value (usually num too big)	18	Not Numeric (VAL func. error)
4	Too Many Variables	19	Program Too Big To Load
5	String Length	20	File Number Invalid
6	No More Data Available For Read	21	Not A SAVEd Program
7	Line Or Input Value >32767	22	'USING' Format
8	Input Or Read Data Error	23	'USING' Too Big
9	Dimension Error	24	'USING' Type
10	Expression Too Complex	25	Dimension Mismatch (RGET)
11	Floating Point Overflow	26	Type Mismatch (RGET)
12	No Such Line Number	27	INPUT Abort
13	NEXT, With No Matching FOR	28	Nesting
14	Line Too Long	29	Player/Missile Number
15	Line Deleted (GOSUB, FOR or WHILE)	30	PM Graphics Not Active
		32	End of 'ENTER'

CIO ERROR MESSAGES

128	Break Abort	133	File Not Open
129	File Number Already Open	134	Bad File Number
130	Nonexistent Device	135	File Is Read Only
131	File Is Write Only	136	End Of File
132	Invalid Command	137	Truncated Record

SIO ERROR MESSAGES

138	Device Timeout	142	Serial Bus Overrun
139	Device NAK (refuses command)	143	Serial Bus Checksum
140	Serial Bus Frame Error		

S: (Screen) ERROR MESSAGE

141	Cursor Out Of Range

HARDWARE ERROR MESSAGES

144	Device Error (usually write protected disk)	145	Read/Write Verify
		146	Invalid Function

SET/SYS VALUES

SET is used to configure certain BASIC A+ system parameters. The companion function SYS( ) may be used to find out what the configuration is at any point in time.

The format is: **SET** parameter number, value. A number in parentheses is the "power-on" default value.

Parameter Number	**Values**	**Meanings**
0	(0)	BREAK key functions normally.
	1	BREAK causes trappable error.
	128	BREAKs are ignored.
1	1 to 127	Tab size for comma in PRINT (10).
2	0 to 255	Prompt character for INPUT (63, "?").
3	(0)	FOR ... NEXT loops execute at least once.
	1	FOR loops may execute zero times.
4	0	Reprompt user if too little INPUT data.
	(1)	No reprompt, a TRAPpable error occurs.
5	0	Lower case/inverse video unchanged.
	(1)	For program entry ONLY, lower case & inverse video converted to upper case.
6	(0)	Print error messages and error numbers.
	1	Print only error numbers.
7	(0)	Player/missiles will NOT wrap around.
	1	Player/missiles wrap around from top to bottom and vice versa.
8	0	No parameter count push for USR calls.
	(1)	DO push the count of parameters.
9	(0)	ENTER statements work like Atari BASIC.
	1	End of an ENTER is treated as a trappable error.

PRINT USING TABLE

Symbol	**Result**
# ...	Blank Fill On Left
* ...	Asterisk Fill On Left
& ...	Zero Fill On Left
,	Numeric Comma Placeholder
.	Numeric Decimal Point Placeholder
$	Fixed Dollar Sign
$ ...	Floating Dollar Sign
+ ...	Floating Forced Sign (+ or −)
− ...	Floating Minus Sign (Blank or −)
% ...	Right Justified String
! ...	Left Justified String
+	Leading Or Trailing Fixed Forced Sign (+ or −)
−	Leading Or Trailing Fixed Minus Sign (Blank or −)
/X	Escape Sequence (X is ANY character and is forced whether in a format or not)